A01 : Broken Access Control (접근 권한 취약점)

접근 권한 취약점이란?
>> 관리자의 의도와 다르게, 해당 계정이 가진 권한 이상의 행동을 할 수 있게 되는 취약점입니다.
대표적인 방법으로 url 직접 접근을 통한 관리자 페이지 노출 취약점이 있습니다.

A01 : 관리자 페이지 노출

```
from urllib.request import *
from urllib.error import *

# 예상되는 관리자 페이지 이름 목록
list = ['admin','administrator',
        'master','manager','management',
        'system','test','anonymous']
text = input('도메인 주소를 입력하세요(www.test.com) : ')

i = 0

# list에 있는 목록을 url에 하나씩 대입한다
for k in list:
    url = 'http://'+text+'/'+k
    try:
        res = urlopen(url) # url 열기
    except HTTPError as e: # HTTP 에러발생 -> 출력하지않기
        continue
    except URLError as e: # URL 에러발생 -> 출력하지않기
        continue
    else: # 예외처리 사항 없음 -> 출력, 취약점 있을때 i값에 1 추가
        i=i+1
        print('\n[',i,'] ',url)

# 관리자 페이지 노출 취약점이 있다면
if i>=1:
    print('\n관리자 페이지 노출 : 취약\n')
else:
    print('\n관리자 페이지 노출 : 양호\n')
```

해당 코드로 관리자 페이지 노출 취약점을 진단했습니다.
예상되는 관리자 페이지 이름을 리스트에 넣은 후, 해당 리스트에 있는 문자열들을 url에 하나씩 대입하는 원리입니다.
에러가 발생하지 않는다면 페이지가 정상 로드된 것으로 간주하며, i변수에 1을 더합니다.

최종적으로 i의 값이 1 이상이라면, 취약하다고 판단합니다.

```
도메인 주소를 입력하세요(www.test.com) : demo.testfire.net

[ 1 ] http://demo.testfire.net/admin

관리자 페이지 노출 : 취약
```

실행 결과
/admin 페이지가 노출되었으며, 따라서 취약한 것으로 진단되었습니다.

A02: Cryptographic Failures(암호화 실패)

암호화 실패란?
>> 암호화와 관련된 전반적인 내용을 다루고 있는 항목이며, 적절한 암호화가 이루어지지 않으면 민감 데이터가 노출될 수 있는 취약점을 말합니다.

취약점 예시는,
데이터가 전송구간에서 평문으로 전송되는 경우(HTTP, FTP, TELNET 등)
취약한 암호화 프로토콜을 사용하는 경우(SSL v2.0, SSL v3.0, TLS v1.0, TLS v1.1)
취약한 암호화 알고리즘을 사용하는 경우(DES, RC4, MD5 등)
취약한 암호화 컴포넌트를 사용하는 경우(취약한 버전의 openssl 사용 등)
보안 헤더 설정을 통한 HSTS가 누락된 경우(HSTS : HTTP를 HTTPS로 강제 리다이렉트)
등등이 있습니다.

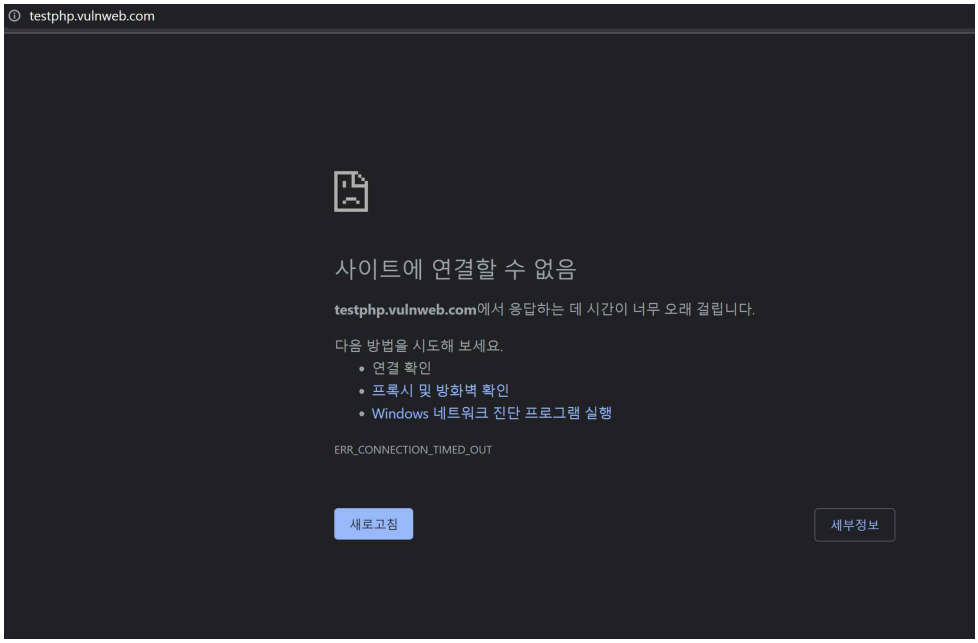
우리는 데이터 평문전송(http 사용)과, HSTS 누락 부분을 기준으로 취약점을 점검하기로 했습니다.

```python
from urllib.error import HTTPError, URLError
from urllib.request import urlopen
from selenium import webdriver

text = input('로그인 주소를 입력하세요(www.test.com) : ')
url = 'https://'+text

try:
    res = urlopen(url)
except HTTPError as e:
    print('취약한 웹 프로토콜 사용')
except URLError as e:
    print('취약한 웹 프로토콜 사용')
else:
    url = 'http://'+text
    driver = webdriver.Chrome("C:/chromedriver.exe")
    driver.get(url)

    variable = driver.current_url

    if variable.lower().startswith('https'):
        print('안전한 웹 프로토콜 사용')
    else :
        print('취약한 웹 프로토콜 사용')
```

점검할 웹 사이트의 주소를 입력받아, https로 접속을 시도합니다.
접속이 되지 않는다면 https를 사용하지 않는다는 것이므로(데이터 평문 전송) 취약한 것으로 간주합니다.

https 접속을 통과했다면 다음 검증으로 넘어갑니다.
이번엔 http로 접속을 시도하여, https로 강제 리다이렉트 되는지 확인합니다.

그렇게 로드가 완료된 사이트의 url을 확인하여
해당 url이 https로 리다이렉트 되었다면 안전한 것으로
해당 url이 https로 리다이렉트 되지 않았다면 취약한 것으로 판단합니다.



testphp.vulnweb.com은 https를 사용하지 않습니다.
위 사진은 https로 직접 접속 시도한 결과입니다.

```
웹 사이트 주소를 입력하세요(www.test.com) : testphp.vulnweb.com
취약한 웹 프로토콜 사용
```

따라서 해당 사이트는 취약하다는 결과가 출력됩니다.

```
웹 사이트 주소를 입력하세요(www.test.com) : demo.testfire.net
C:\Users\ysk95\PycharmProjects\취약점프로젝트\a02완성품.py:17: Deprec
  driver = webdriver.Chrome("C:/chromedriver.exe")
취약한 웹 프로토콜 사용
```

위 사진은 https를 사용하지만, 자동 리다이렉트는 사용하지 않는 사이트입니다.
따라서 2번째 검증에서 필터링 되어 취약하다는 결과가 출력됩니다.

```
웹 사이트 주소를 입력하세요(www.test.com) : www.google.com
C:\Users\ysk95\PycharmProjects\취약점프로젝트\a02완성품.py:17: Depre
  driver = webdriver.Chrome("C:/chromedriver.exe")
안전한 웹 프로토콜 사용
```

www.google.com과 같이 안전한 웹 사이트는 안전하다는 결과가 출력됩니다.

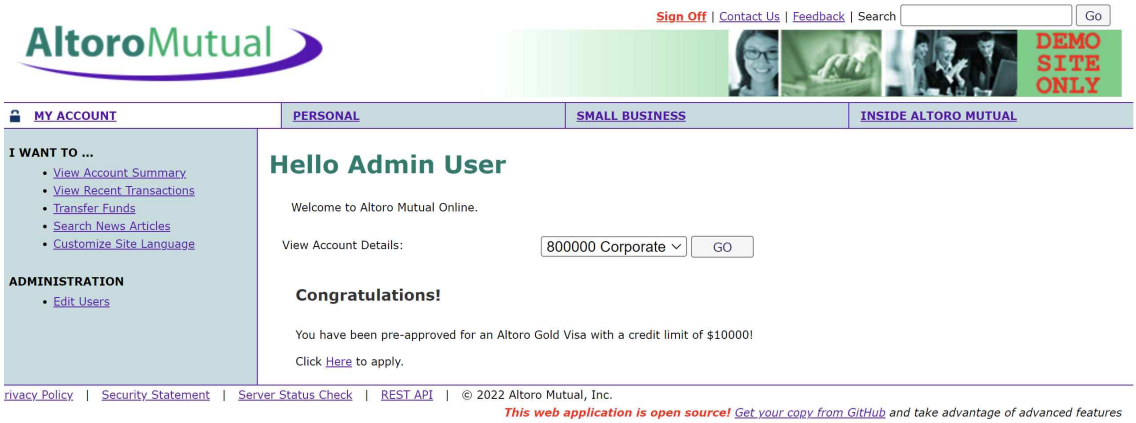
A03 : Injection(인젝션)

Injection이란?
>> 악의적인 사용자가 보안상의 취약점을 이용하여, 임의의 SQL 문을 주입하고 실행되게 하여 데이터베이스가 비정상적인 동작을 하도록 조작하는 행위입니다. 공격이 비교적 쉬운 편이고 공격에 성공할 경우 중요 데이터를 탈취하는 등 큰 피해를 입힐 수 있는 공격입니다.

A03 : SQL Injection

```
# 인증우회 sql injection을 사용해 SQLi에 취약한지 진단하는 코드
from selenium import webdriver

# 점검을 시도할 url 입력
text = input('로그인 주소를 입력하세요(www.test.com) : ')
url = 'http://'+text

xpath_id = '//*[@id="uid"]'
xpath_pw = '//*[@id="passw"]'
xpath_button = '//*[@id="login"]/table/tbody/tr[3]/td[2]/input'

# id는 admin으로 가정
id = "admin"
pw = "' or '1'='1"

driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(url)

driver.find_element("xpath", xpath_id).send_keys(id)
driver.find_element("xpath", xpath_pw).send_keys(pw)

driver.find_element("xpath", xpath_button).click()

variable = driver.current_url #현재 페이지 출력 (로그인 성공 여부 확인 가능)

# 페이지가 정상출력이 된다면( 인증우회 SQL Injection으로 로그인이 되었다면 )
if variable.lower().startswith('http'):
    print('sql_injection : 취약')
else:
    print('sql_injection : 양호')
```

해당 코드를 통하여 임의의 웹 페이지에 인증우회 SQL Injection 시도를 하였습니다.
'or 1=1 코드를 통해(참이되는 값 전송) 로그인이 되는지 확인해 보도록 하겠습니다.

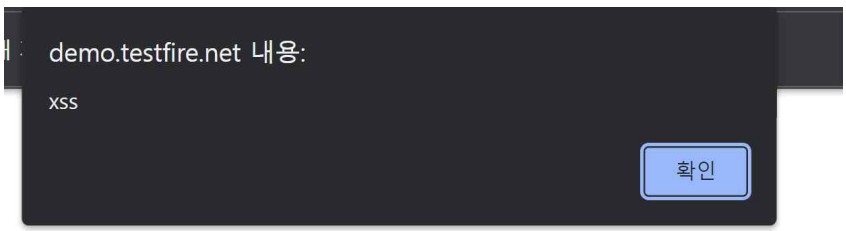


‘ or 1=1 코드로 관리자 계정으로 로그인이 성공했습니다.
따라서 해당 웹 사이트는 취약한 것으로 진단되었습니다.

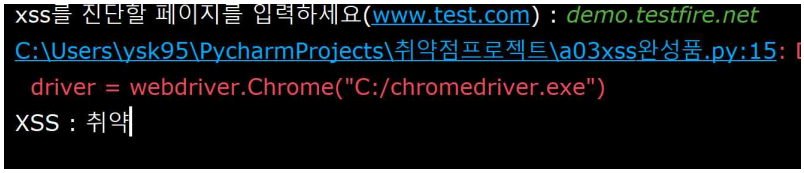
A03 : XSS(크로스 사이트 스크립트)

```
# xss취약점 진단
from selenium import webdriver
from selenium.webdriver.common.alert import Alert

text = input('xss를 진단할 페이지를 입력하세요(www.test.com) : ')
url = 'http://'+text

# 스크립트를 입력할 파라미터( 입력할 위치 )
xpath_text = '//*[@id="query"]'
xpath_button = '//*[@id="frmSearch"]/table/tbody/tr[1]/td[2]/input[2]'

# 입력할 스크립트문
script = "<script>alert('xss');</script>"

driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(url)

# 스크립트문 입력 후 전송
driver.find_element("xpath", xpath_text).send_keys(script)
driver.find_element("xpath", xpath_button).click()

# 스크립트문이 동작하여, alert창이 떴을 경우
try:
    Alert(driver).accept()
    print('sql_injection : 취약')
except:
    print('sql_injection : 양호')
```

해당 코드로 진단을 실행하였습니다.
웹 사이트 검색엔진 등 입력창에 스크립트문을 입력한 후,
해당 스크립트문이 동작하면 취약한 것으로 판단합니다.
해당 코드에서는 alert문을 이용하였으며, 따라서 alert창이 출력되면 취약하다고 판단합니다.



alert창 출력



따라서 취약하다는 결과가 출력되었습니다.

A04 : Insecure Design (안전하지 않은 설계)

안전하지 않은 설계란?
>> 안전하지 않은 설계 누락되거나 비효율적인 제어 설계로 표현되는 다양한 취약점을 나타냅니다. 안전하지 않은 설계와 안전하지 않은 구현에는 차이가 있지만, 안전하지 않은 설계에서 취약점으로 이어지는 구현 결함이 있을 수 있습니다.

해당 부분을 점검,예방하는 방법입니다.
- 요구사항 및 리소스 관리
  모든 데이터 자산과 예상되는 비즈니스 로직의 기밀성, 무결성 가용성 및 신뢰성에 관한 보안 요구사항을 포함하여 기획
- 보안 설계
  알려진 공격 기법을 방지하기 위해 위협을 지속적으로 평가하고 코드가 안전하게 설계되고 테스트됐는지 확인하는 문화 및 방법론 적용
- 보안 개발 생명 주기
  전체 프로젝트 및 소프트웨어 유지 관리의 전반에 걸쳐 프로젝트 시작 단계에서부터 보안 전문가의 검증 필요

보기와 같이, A04번은 광범위하며 판단 기준이 모호합니다.
따라서 이를 파이썬 코드로 구현하는 것은 어렵다고 판단하였습니다.

A05 : Security Misconfiguration (보안설정오류)

보안설정오류란?
>> 애플리케이션 스택의 적절한 보안 강화가 누락되었거나 클라우드 서비스에 대한 권한이 적절하지 않게 구성되었을 때, 불필요한 기능이 활성화 되거나 설치되었을 때, 지나치게 상세한 오류 메세지를 노출할 때, 최신 보안기능이 비활성화 되거나 안전하지 않게 구성되었을 때 발생합니다.

해당 부분에서는 지나치게 상세한 오류 페이지를 취약점 기준으로 잡고 작업하였습니다.

```
from selenium import webdriver

from urllib.request import * # 웹페이지 요청 + 데이터 가져오기
from urllib.error import * # 존재하는 url인지 확인

error_list1 = ['No results were found for the query', 'syntax error', '로 변환하지 못했습\
니다','변환하는 중 구문 오류가 발생했습니다.','따옴표가 짝이 맞지 않습니다.','You have \
an error in your SQL syntax','Unclosed quotation mark after the character string','Orac\
le Text error:'] # SQLi 오류 페이지 데이터 유출 확인을 위한 문자열

text = input('에러페이지 확인 주소 입력(www.test.com) : ')
url = 'http://'+text

# SQLi 코드를 입력할 위치
xpath_text = '//*[@id="query"]'
xpath_button = '//*[@id="frmSearch"]/table/tbody/tr[1]/td[2]/input[2]'

# SQLi 코드
script = """

driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(url)

driver.find_element("xpath", xpath_text).send_keys(script)
driver.find_element("xpath", xpath_button).click()
```

```
html = driver.page_source

summ = 0

# 웹페이지에 오류구문이 있다면 summ += 1
for i in error_list1:  # 오류구문 하나씩 검증
    if i in html:
        summ += 0

if summ != 0:   # 오류구문이 발견되었다면
    print('에러페이지 정보노출 : 취약')
# SQLi를 안전으로 통과하면 else문(웹 페이지 오류노출)으로 진행
else:
    error_list2 = ['Apache', 'nginx', 'IIS'] # 해당 문자열이 있으면 취약으로 판단
    url = url+'/errorpage_check' # 오류를 유발할 주소

    driver = webdriver.Chrome("C:/chromedriver.exe")
    driver.get(url)

    try:
        res = urlopen(url) # url 열기
    except HTTPError as e: # HTTP 에러발생 -> 웹서버 정보 나오는지 검증
        summ == 0
        html = driver.page_source
        for j in error_list2:
            if j in html:  # 웹페이지 오류에서 정보노출이 된다면
                print('에러페이지 정보노출 : 취약')
                break
            else:   # SQLi, 웹페이지 모두 안전하다면 안전 출력
                summ += 1

if summ==int(len(error_list2)):
    print('에러페이지 정보노출 : 안전')
```

해당 코드에는 2개의 검증구문이 있습니다.
첫번째는 SQLi 를 시도하였을 때 나오는 에러페이지에서, DB정보가 노출되는 등 과도한 정보가 출력되면 취약한 것으로 간주하였고, SQLi가 안전할 때 두 번째 검증구문으로 넘어갑니다.

두번째는 고의로 존재하지 않는 위치를 url에 입력하여 HTTP 에러페이지를 띄운 후,
해당 페이지에서 apache tomcat이나 nginx 등 서버정보가 출력되는지 확인하였습니다.
서버정보가 한 개라도 노출되고 있다면 취약으로 판단하고,
최종적으로 모두 안전하다면 '에러페이지 정보노출: 안전' 을 출력합니다.

```
에러페이지 확인 주소 입력(www.test.com) : demo.testfire.net
C:\Users\ysk95\PycharmProjects\취약점프로젝트\a05완성품.py:23: DeprecationW
  driver = webdriver.Chrome("C:/chromedriver.exe")
C:\Users\ysk95\PycharmProjects\취약점프로젝트\a05완성품.py:46: DeprecationW
  driver = webdriver.Chrome("C:/chromedriver.exe")
에러페이지 정보노출 : 안전
```

프로그램 실행 결과입니다.
해당 사이트는 SQLi 에러 페이지에서 정보 노출이 되지 않고 있으며, HTTP 에러 페이지에서 서버정보를 노출하지도 않습니다.
따라서 '에러페이지 정보노출 : 안전' 을 출력합니다.

A06: Vulnerable and Outdated Components
(취약하고 오래된 요소)

취약하고 오래된 요소란?

>> 취약하고 오래된 요소는 지원이 종료되었거나 오래된 버전을 사용할 때 발생합니다. 이는 애플리케이션 뿐만 아니라 os, 커널, 소프트웨어, 워드프레스의 플러그인 등 굉장히 다양한 곳에서 발생할 수 있습니다.

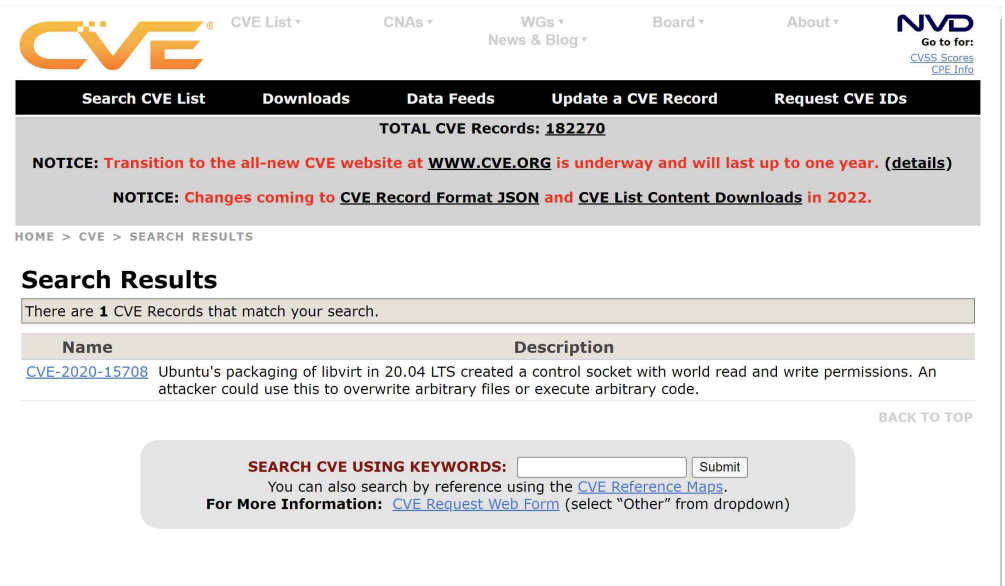
```python
from selenium import webdriver #크롤링을 위한 모듈
from bs4 import BeautifulSoup

# 취약점을 검색할 사이트
text = 'https://cve.mitre.org/cve/search_cve_list.html'

# 해당 사이트의 검색 파라미터
xpath_search = '//*[@id="CenterPane"]/form/div[1]/input'
xpath_button = '//*[@id="CenterPane"]/form/div[2]/input'

# 점검할 서비스의 이름과 버전을 입력받는다
search = input('점검할 서비스 입력 ( ex : ubuntu 20.04 ) : ')

#크롬 브라우저 실행 -> url 열기
driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(text)

#지정한 파라미터에 사용자 입력값 전송
driver.find_element("xpath", xpath_search).send_keys(search)
driver.find_element("xpath", xpath_button).click()

# 검색 결과 페이지 소스코드
html = driver.page_source
soup = BeautifulSoup(html,'html.parser')

# td태그를 전부 가져옴
td_tag = soup.find_all('td')

count = 0
```

```
# td태그의 갯수 세기
for i in td_tag:
    count+=1

# 취약점이 1개 이상 있다면( 취약점 1개당 td2개 증가, 기본값 19개 )
if int(count)>=21:
    print('버전 업데이트 또는 변경이 필요합니다.')
else:
    print('안전합니다.')
```

해당 코드는 각종 취약점을 검색할 수 있는 cve 사이트를 이용한 코드입니다.
사용자가 점검할 서비스의 이름과 버전을 입력하면, 프로그램이 cve에 검색을 한 후
취약점이 1개 이상 발견될 시 '버전 업데이트 또는 변경이 필요합니다.' 를 출력합니다.

우리는 취약점 유무를 td태그를 이용하여 판단하기로 했습니다.

해당 웹 페이지에서 취약점 목록은 td 태그를 이용하여 표현되며, 취약점 1개당 html코드의 td 갯수가 2개씩 늘어나게 됩니다.
기본값 td 개수는 19개이므로, 21개라면 취약점 1개가 있는 것입니다.
따라서 td가 21개 이상이라면 사용자에게 취약하다는 결과를 출력합니다.



ubuntu 20.04를 검색한 결과.
취약점 1개가 발견된 모습

```
점검할 서비스 입력 ( ex : ubuntu 20.04 ) : ubuntu 20.04
C:\Users\ysk95\PycharmProjects\취약점프로젝트\a06완성품.py:16: Dep
  driver = webdriver.Chrome("C:/chromedriver.exe")
버전 업데이트 또는 변경이 필요합니다.
```

따라서 업데이트가 필요하다는 문구가 출력됩니다.

A07 : Identification and Authentication Failures
(식별 및 인증 오류)

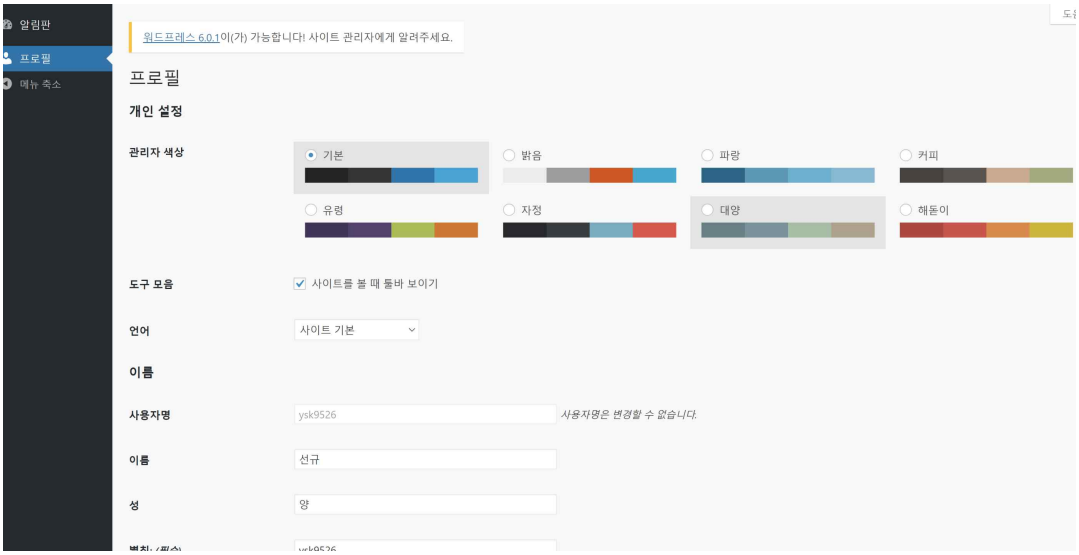
식별 및 인증 오류란?
>> 사용자의 신원확인, 인증 및 세션관리가 적절히 되지 않을 때 취약점이 발생할 수 있습니다.
해당되는 취약점은 인증 실패에 대한 제한이 없어 Brute forcing 공격에 노출되는 경우,
안전한 비밀번호 생성 정책이 없어, 취약한 비밀번호 생성을 허용하는 경우
세션 타임아웃이 없거나 로그아웃 후 세션 파기를 하지 않는 경우 등등이 있습니다.

우리는 그 중 인증 실패에 대한 제한이 없어 Brute force 공격을 허용하는 경우를 기준으로 삼았습니다.

```python
from itertools import product
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from urllib.error import *
from urllib.request import *
import sys

# 조합을 만들 문자열 목록
words ='wyjk'

# id,pw, 로그인 버튼 파라미터
xpath_id = '//*[@id="user_login"]'
xpath_pw = '//*[@id="user_pass"]'
xpath_click = '//*[@id="wp-submit"]'

# 로그인 페이지 주소와 id를 입력받는다
text = input('로그인 주소를 입력하세요(www.test.com) : ')
id = input('ID : ')

url = 'http://' + text
error = 0
```

```python
for passwd_length in range(2,3): # 시도할 패스워드 자릿수 설정
    admin_passwd = product(words, repeat=passwd_length) #words 문자열을 조합해 대입한다
    for passwd_tmp in admin_passwd:
        passwd = ''
        passwd = ''.join(passwd_tmp) #passwd엔 조합된 문자열이 들어가 있다
        driver = webdriver.Chrome(ChromeDriverManager().install())
        driver.get(url)
        driver.find_element('xpath',xpath_id).send_keys(id)
        driver.find_element('xpath',xpath_pw).send_keys(passwd)
        driver.find_element('xpath',xpath_click).click()

        try:  # 로그인 시도한 결과 소스코드를 가져온다
            res = urlopen(driver.current_url)
            html = driver.page_source

        except HTTPError as e: # 로그인 실패 시 이어서 반복한다
            continue
        else:
            if "안녕하세요" in html:  # 로그인 성공 시 나오는 문자열을 검증
                print('Brute Force : 취약')  # 로그인이 성공했다면 취약
                error = 1
                sys.exit()
            else:
                continue
# 최종적으로 로그인에 성공하지 못했다면
if error != 1:
    print('Brute Force : 양호')
```

해당 코드는 임의의 문자열을 만들어 입력받은 사이트에 Brute force 공격을 시도하는 코드입니다.
로그인 시도 후, 로그인 성공을 의미하는 문자열이 html코드에 존재한다면 취약하다고 판단했습니다.
주어진 문자열 조합으로 끝까지 로그인에 성공하지 못했다면 '양호'를 출력하고,
로그인에 성공했다면 '취약' 을 출력합니다.



프로그램 실행 후, 로그인 성공

Brute Force 공격으로 로그인에 성공했으므로, 취약하다는 결과를 출력합니다.

A08 : Software and Data Integrity Failures
(소프트웨어 및 데이터 무결성 오류)

소프트웨어 및 데이터 무결성 오류란?
>> 소프트웨어 및 데이터 무결성 오류는 애플리케이션이 신뢰할 수 없는 소스, 저장소 및 CDN, 플러그인, 라이브러리, 모듈에 의존하는 경우에 발생합니다.

취약점 예시는
애플리케이션이 사용하는 라이브러리나 모듈에 대한 무결성 검증이 없어 변조가 가능한 경우
업데이트 공급망에 대한 검증이 없는 경우
CI/CD 파이프라인에 대한 적절한 보안성 검토가 없는 경우
직렬화된 데이터에 대한 무결성 검증이 없는 경우
등이 있습니다.

우리는 웹 서버와 클라이언트가 통신할 때, 웹에서 해시값을 함께 주어서 클라이언트가 무결성을 검증할 수 있는지를 진단하려고 했습니다.
그러나 세션값은 제3자가 가지고 올 수 없는 값이었고, 웹 사이트 response 해시값을 받아오는 방법은 찾지 못했습니다.
따라서, 파이썬 코드로 개발이 힘들다고 판단하였습니다.

A09 : Security Logging and Monitoring Failures
(보안 로깅 및 모니터링 오류)

보안 로깅 및 모니터링 오류란?
>> 적절한 로깅과 모니터링이 없다면 공격을 감지하고 대응할 수가 없습니다. 취약점 공격 예방과 공격 감지, 그리고 대응까지 포함하는 항목입니다.

취약점 예시는
로그인, 인증 실패, 권한 설정 등 중요 기능 수행에 대한 로깅이 없는 경우
일정 주기로 로그에 대한 백업 절차가 없는 경우
등이 있습니다.

이를 진단하기 위해선, 웹 서버 내 로깅은 어떻게 이루어지고 있는지, 로그는 어떻게 어디에 백업하고 있는지를 알아야 합니다.

하지만 이는 서버 내에 있어야 하고, 높은 권한이 있어야 알 수 있는 부분이라고 생각했으며, 실제 모니터링을 어떻게 하고 있는지는 웹 상에서 제3자가 판단할 수 없기 때문에
파이썬 코드로는 구현이 어렵다고 판단하였습니다.